PATENT APPLICATION
ATTORNEY DOCKET NO. 15115.012001

APPLICATION

FOR

UNITED STATES LETTERS PATENT

FOR TTC 15115.012001

TITLE: **METHOD AND APPARATUS FOR DELETING DATA**

APPLICANT: **Taiichi YUASA**

"EXPRESS MAIL" Mailing Label Number: EV042549181US
Date of Deposit: November 14, 2001



22511
PATENT TRADEMARK OFFICE

METHOD AND APPARATUS FOR DELETING DATA

Background of Invention

[0001] 本発明は、主プログラムの処理と交互に、主プログラムの実行に使用されるデータを記録しているデータ記録領域を走査し、不要となったデータを消去するデータ消去方法、その方法を適用したデータ消去装置及びその装置を実現するためのプログラムが記録されている記録媒体に関し、特に動的にデータを生成するプログラミング言語にて作成された実時間性を必要とする応用ソフトウエアプログラムにて発生する不要なデータを消去するデータ消去方法、データ消去装置及び記録媒体に関する。

[0002] Ｊａｖａ、Ｃ＋＋、Ｌｉｓｐ及びＰｒｏｌｏｇ等の動的にデータを生成するプログラミング言語を用いて作成された応用ソフトウエアプログラム（以下主プログラムという）では、主プログラムの処理中に一度生成され使用されたデータが、主プログラムの処理が進むにつれて不要になる場合がある。

[0003] そこでデータを記録するデータ記録領域を有効に利用するため、これらの不要データをデータ記録領域から消去し、データを消去した後のデータ記録領域を他のデータの記録に再利用する必要がある。 このようにデータを記録し、再利用する処理は「ゴミ集め」（Garbage Collection：以下ＧＣという）と呼ばれ、主プログラムの処理に不可欠な処理である。

[0004] 実時間性を重要視する主プログラムを処理する場合、ＧＣは主プログラムの処理に対し、実質的に並行して実行する必要があるプログラムであり、その方法として、On-the-flyＧＣ、複写方式インクリメンタルＧＣ及びスナップショットＧＣ等の方法が提案され、また実用化されている。

[0005] On-the-flyＧＣは、主プログラムを処理するプロセッサとは別にＧＣプログラム専用のプロセッサを用意し、２つのプロセッサを並行に動作させて実時間でＧＣプログラムを実行する方法であり、主プログラムによりデータ記録領域を指示するポインタの指示を監視しながら、ポインタにより指示されているデータ記録領域にマーキングを行い、マーキングされなかった不要なデータ記録領域のデータを消去する Dijkstra によって考案されたアルゴリズム及び圧縮処理を用いた Steele によって考案されたアルゴリズム等の処理を適用した方法である。

[0006] 複写方式インクリメンタルＧＣは、使用可能な記録領域から２つのデータ記録領域（from 空間及び to 空間）を確保して、一方のデータ記録領域（from 空間）のみを用いて主プログラムの処理を実行し、所定のタイミングで、ポインタにより指示される有用なデータを他方のデータ記録領域（to 空間）に複写した上で、一方のデータ記録領域のデータを消去する Baker によって考案されたアルゴリズムを適用した方法である。

[0007] ところが On-the-flyＧＣ及び複写方式インクリメンタルＧＣのいずれにおいても、プログラムの実行に大きなオーバーヘッドが生じ、しかもプロセッサ及び記録領域等のハードウエア資源の追加又は確保が必要になり、汎用機での実用は困難であった。

[0008] これに対し本発明の発明者である湯淺によって考案されたスナップショットＧＣは、Dijkstra 同様にデータ記録領域にマーキングを行い消去するアルゴリズムをベースにしながらも、汎用機で容易に実現することを目的としたアルゴリズムであり、その優秀さから現状では最も広く用いられている。

[0009] スナップショットＧＣでは、主プログラムの処理に必要な関数の関数記録領域及び該関数記録領域以外の予め場所が決まっている静的記録領域を、主プログラムの処理を中断して走査し、走査した関数記録領域に記録されているポインタから直接又は間接的に指示されているデータ記録領域にデータの保護を予約する予約マーキング（マーキングの予約）を行い、更に予約マーキングされた記録領域のデータを保護すべくマーキングを行う。そしてマーキングされていないデータ記録領域に記録されたデータを消去するアルゴリズムを適用した方法である。

[0010] このようなスナップショットＧＣの処理を、図を用いて説明する。図１０は主プログラムの実行に必要な関数の関数記録領域を概念的に示す説明図である。主プログラムは、一般に関数の呼び出しが入れ子状態になって実行される。即ち図１０（ａ）、（ｂ）及び（ｃ）に示すように主プログラムの処理に必要な関数は、実行中の関数から実行すべき関数を呼び出して、メモリ中のスタック領域に確保された実行中の関数の関数記録領域に、新たな関数記録領域を積み上げる形で確保して、呼び出された関数を実行し、関数の実行完了後、積み上げられている不要になった関数記録領域を破棄するようにして処理される。

[0011] 実行中の関数の関数記録領域（関数フレーム）は、カレントフレームと呼ばれ、図１０（ａ）では、関数Ｆが実行中であり、関数Ｆの関数記録領域がカレントフレームとなる。この関数Ｆから別の関数Ｇが呼び出されると、図１０（ｂ）に示すように、関数Ｇを実行するための関数記録領域がスタック領域に積み上げられて、カレントフレームとなり、関数Ｇの実行が完了すれば、当該関数記録領域は破棄されて、呼び出し前の図１０（ａ）の状態となる。

[0012] なお図１０（ｂ）において、さらに他の関数Ｈが呼び出されたときは、図１０（ｃ）に示すように、関数Ｈを実行するための関数記録領域がスタック領域に積み上げられて、カレントフレームとなり、関数Ｈの実行が完了すれば、当該関数記録領域は破棄されて、呼び出し前の図１０（ｂ）の状態となる。

[0013] 図１１は従来のデータ消去方法（スナップショットＧＣ）の走査及びマーキング処理を概念的に示す説明図である。なお図１１において、左下がりの斜線部は、走査済みの関数記録領域を示し、右下がりの斜線部は、未走査の関数記録領域を示す。関数記録領域に記録されているポインタの走査は、実行中である最上位の関数記録領域から、基幹の関数となる最下位の関数記録領域へ向かって行われ、走査したポインタが指示するデータ記録領域に予約マーキングを行い、そして予約マーキングが行われたデータ記録領域にマーキングを行い、マーキングされていないデータ記録領域に記録されたデータを不要なデータとみなして消去する。

[0014] ところでスナップショットＧＣの処理の中で、静的記録領域の走査及び予約マーキングについては、静的記録領域に記録されているポインタが少数であるため短時間で完了するが、関数記録領域の走査及び予約マーキングは、走査すべき関数記録領域が増減するため、必ずしも短時間で完了するわけではない。

[0015] しかしながらスナップショットＧＣは、主プログラムを中断してその処理を行うため、実時間性を維持してＧＣプログラムを実行するためには、ＧＣに関する処理を分割して行わなければならない。

[0016] 静的記録領域の走査及び予約マーキングを行う処理は、短時間で完了するため主プログラムが中断している間に実行することが可能であり、

また予約マーキングされたデータ記録領域のマーキング及びデータを消去する処理は、短時間で完了しなくとも、分割して処理することが可能であるため、これらの処理はいずれも大きな問題とはならない。

[0017] ところが関数記録領域の走査及び予約マーキングは、処理が長時間になる可能性があるため、実時間性が損なわれる可能性があるという問題がある。

[0018] そこで関数記録領域の走査及び予約マーキングの処理を分割し、主プログラムの中断毎に少しずつ走査して１回の中断時間を短縮し、主プログラムの実時間性を維持することが考えられるが、この場合、以下に例示するような処理異常が発生する可能性がある。

[0019] 図１２は、従来のデータ消去方法（スナップショットＧＣ）の走査及びマーキング処理を概念的に示す説明図である。なお図１１と同様に、左下がりの斜線部は、走査済みの関数記録領域を示し、右下がりの斜線部は、未走査の関数記録領域を示す。また図１２（ａ）に示す状態で、ＧＣ処理が中断されて、主プログラムが実行され、図１２（ｂ）に示す状態に遷移したと仮定する。図１２（ａ）に示す状態では、データ記録領域ａは予約マーキングされているが、データ記録領域ｂ及びｃは予約マーキングされていない状態である。

[0020] 図１２（ａ）に示す状態から主プログラムが実行されて、図１２（ｂ）に示す状態に遷移した場合、データ記録領域ｂは指示されてはいるが、データ記録領域ｂを指示するポインタを記録している関数記録領域が、未走査の関数記録領域から走査済みの関数記録領域に変化しており、ＧＣ処理が再開されたときに、データ記録領域ｂには、予約マーキングが行われていないため（予約マーキングの取りこぼしが発生しているため）、デ

ータが保護されずに消去される。このため主プログラムの実行により、データ記録領域ｂを必要とする処理が行われる場合、そのデータが消去されていることにより処理異常が発生する。

[0021] このような取りこぼしが発生する状況を説明する。図１３は、従来のデータ消去方法（スナップショットＧＣ）の走査及びマーキング処理を概念的に示す説明図である。一般に、関数の実行中は、関数記録領域全てが参照されるのではなく、カレントフレーム内だけが参照される。

[0022] 図１３（ａ）では、カレントフレームが走査済みの関数記録領域に含まれている状態を示しており、このような状態の場合、主プログラムの処理によりスタックの状態が変化しても、影響を受けるのは走査済みの関数記録領域内だけであるため、取りこぼしが生じることはない。

[0023] 図１３（ｂ）では、カレントフレームが未走査の関数記録領域に含まれている状態を示しており、主プログラムの処理によりスタックの状態が変化して、特定のデータ記録領域を指示するポインタが、走査済みの関数記録領域に移動し、取りこぼしが生じる可能性がある。

[0024] また上述したような基本的な処理を実行中に発生する処理異常だけでなく、局所関数（Local functions）をサポートする言語システムの場合は、以下に例示するような状況を考慮しなければならないという課題がある。

[0025] 図１４は、従来のデータ消去方法（スナップショットＧＣ）の走査処理を概念的に示す説明図である。 図１４では、局所関数を含んだ以下の簡単な Common Lisp コードを想定している。

[0026] (defun F(x)

(labels((G(y z)...(G z y)...))

(G x x)))

[0027] 即ち関数Ｆの中で、局所関数Ｇが定義されており、関数Ｇからは関数Ｇ自身が再帰的に呼び出されている。また再帰呼び出しの式に現れる変数ｘは、関数Ｆにて定義されているパラメータであり、関数Ｆの関数記録領域中に存在する。このため関数Ｇの実行中は、関数Ｇの関数記録領域だけでなく、該関数記録領域より下位の関数Ｆの関数記録領域をも参照することになり、一般的には関数Ｇの関数記録領域に、関数Ｆの関数記録領域の参照を可能とする静的リンク（static link）を格納するということにより実現されている。この場合、図１４に示すように、既に走査された関数Ｇの関数記録領域から、走査されていない関数Ｆの関数記録領域を参照するという状況が発生し、様々な不都合が生じかねない。

[0028] さらに主プログラムの実行の中断等の割込命令による例外処理又は非局所的脱出により、実行する関数が、実行中の関数から、該関数を呼び出した関数ではない下位の戻り先となる関数となる状況をも考慮しなければならないという課題もある。

[0029] 図１５は、従来のデータ消去方法（スナップショットＧＣ）の走査処理を概念的に示す説明図である。考慮すべき状況とは、図１５（ａ）に示すように関数Ｈを実行中に、例外処理又は非局所的脱出が行われることにより、関数Ｈの実行を終了させた後、関数Ｈを呼び出した関数Ｇではなく、図１５（ｂ）に示すように更に下位の関数Ｆを戻り先の関数として実行するthrowと呼ばれる機能が実行される状況である。

[0030] throwが実行される場合において、関数Ｈの関数記録領域は既に走査されているが、関数Ｇの関数記録領域及び関数Ｆの関数記録領域は走査

されていないという状況が発生することもあり、そのときの処理を如何にするかが、例えば非局所的脱出をサポートする言語システムを実装する上での課題として残ることになる。

Summary of Invention

[0031] 本発明は斯かる事情に鑑みてなされたものであり、スナップショットＧＣの走査及び予約マーキングを中断する場合に、走査済みの関数記録領域に障壁を設定し、障壁が設定された関数記録領域が破棄されるときには、主プログラムの処理より走査及び予約マーキングを優先して行うことにより、予約マーキングの取りこぼしによる処理異常の発生を防止するデータ消去方法、その方法を適用したデータ消去装置及びその装置を実現するためのプログラムが記録されている記録媒体の提供を主たる目的とする。

[0032] さらに本発明では、走査を行わせる関数の番地を、障壁として設定することによりオーバーヘッドを抑制し、主プログラムの修正が不要であるデータ消去装置等の提供を他の目的とする。

[0033] また本発明では、局所関数をサポートする言語システムの場合において、走査すべき関数記録領域にて実行される関数が、下位の関数記録領域を参照しているときに、参照元の関数記録領域だけでなく参照先の関数記録領域をも走査することにより、例えば障壁が参照元の関数記録領域及び参照先の関数記録領域の間に設定されていたとしても処理異常の発生を防止し、しかも走査すべき関数記録領域を実質的に少なくすることができるので実時間性を損なうことがないデータ消去装置等の提供を更に他の目的とする。

[0034] そして本発明では、実行する関数が、実行中の関数から、該関数を呼び出した関数ではない更に下位の戻り先となる関数に移行する throw が実行される場合において、実行中の関数の関数記録領域と戻り先となる関数の関数記録領域との間に障壁が設定されているときに、戻り先となる関数の関数記録領域から走査を開始することにより、走査すべき関数記録領域を実質的に少なくすることができるので実時間性を損なうことがなく、しかも実行中の関数の関数記録領域及び戻り先となる関数の関数記録領域との間の走査されていない関数記録領域については、参照されることがないので走査を行わなくとも処理異常は発生しないデータ消去装置等の提供を更に他の目的とする。

[0035] 第１発明に係るデータ消去方法は、メモリ中のスタック領域に、関数の実行に必要な関数記録領域を積み上げて関数を実行し、実行完了後、当該関数記録領域を破棄する主プログラムの処理と交互に、関数の実行に使用されるデータを記録するデータ記録領域に記録された保護されていない不要なデータを消去するデータ消去方法において、スタック領域に積み上げられた関数記録領域を上位側から下位側へ走査し、走査された関数記録領域に記録されているポインタが指示するデータ記録領域のデータの保護を予約する予約マーキングを行い、主プログラムの処理を実行すべく、関数記録領域の走査を中断する場合に、走査が最後に行われた関数記録領域に、関数の実行を制限する障壁を設定し、該障壁が、関数の実行が完了して破棄されるべき関数記録領域に設定されているとき、破棄より走査を優先させることを特徴とする。

[0036] 第１発明に係るデータ消去方法では、走査済みの関数記録領域に障壁を設定して、障壁が設定された関数記録領域が破棄されるときに、主プ

ログラムの処理より走査及び予約マーキングを優先して行うことにより、関数記録領域が破棄されてカレントフレームが未走査の関数記録領域に移行することを防止するので、走査処理を中断している間の主プログラムの処理に基づくポインタの変化により発生する取りこぼしを防止して、必要なデータの消去を回避し、処理異常の発生を防止して、主プログラム処理時の実時間性を損なわずにデータ記録領域に記録された不要なデータの消去が可能である。

[0037] 第２発明に係るデータ消去装置は、メモリ中のスタック領域に、関数の実行に必要な関数記録領域を積み上げて関数を実行し、実行完了後、当該関数記録領域を破棄する主プログラムの処理と交互に、関数の実行に使用されるデータを記録するデータ記録領域に記録された保護されていない不要なデータを消去するデータ消去装置において、スタック領域に積み上げられた関数記録領域を上位側から下位側へ走査する走査手段と、走査された関数記録領域に記録されているポインタが指示するデータ記録領域のデータの保護を予約する予約マーキングを行う手段と、主プログラムの処理を実行すべく、関数記録領域の走査を中断する場合に、走査が最後に行われた関数記録領域に、関数の実行を制限する障壁を設定する手段と、該障壁が、関数の実行が完了して破棄されるべき関数記録領域に設定されているとき、破棄より走査を優先させる手段とを備えることを特徴とする。

[0038] 第２発明に係るデータ消去装置では、走査済みの関数記録領域に障壁を設定して、障壁が設定された関数記録領域が破棄されるときに、主プログラムの処理より走査及び予約マーキングを優先して行うことにより、関数記録領域が破棄されてカレントフレームが未走査の関数記録領域に移行することを防止するので、走査処理を中断している間の主プログラムの

実行に基づくポインタの変化により発生する取りこぼしを防止して、必要なデータの消去を回避し、処理異常の発生を防止して、主プログラム処理時の実時間性を損なわずにデータ記録領域に記録された不要なデータの消去が可能である。

[0039] 第３発明に係るデータ消去装置は、第２発明において、前記障壁は、走査を行わせる関数を呼び出す手段であることを特徴とする。

[0040] 第３発明に係るデータ消去装置では、走査を行わせる関数の番地を障壁として設定することにより、関数記録領域の破棄より走査を優先させる処理のための主プログラムの修正が不要であり、しかもその処理によるオーバーヘッドを抑制するので、主プログラム処理時の実時間性を損なわずにデータ記録領域に記録された不要なデータの消去が可能である。

[0041] 第４発明に係るデータ消去装置は、第２発明又は第３発明において、走査する第１の関数記録領域の関数が、前記第１の関数記録領域より下位の第２の関数記録領域を参照しているか否かを判定する手段を備え、第２の関数記録領域を参照していると判定した場合、前記走査手段は、第１の関数記録領域を走査するときに、第２の関数記録領域も走査すべくなしてあることを特徴とする。

[0042] 第４発明に係るデータ消去装置では、第１の関数記録領域にて実行する関数が、第２の関数記録領域を参照している場合で、第１の関数記録領域を走査するときに、第２の関数記録領域をも走査することにより、障壁が第１の関数記録領域及び第２の関数記録領域の間に設定されていたとしても処理異常の発生を防止し、しかも走査すべき関数記録領域を実質的に少なくすることができるので実時間性を損なうことがない。

[0043] 第５発明に係るデータ消去装置は、第２発明乃至第４発明のいずれかにおいて、実行する関数が、実行中の関数から、該関数の実行に必要な関数記録領域を積み上げさせた関数と異なる下位の戻り先関数に戻る場合、前記実行中の関数の関数記録領域及び戻り先関数の関数記録領域の間に、障壁を設定された関数記録領域の有無を判定する手段を備え、障壁を設定された関数記録領域が有ると判定したときに、前記走査手段は前記戻り先関数の関数記録領域から走査すべくなしてあることを特徴とする。

[0044] 第５発明に係るデータ消去装置では、実行中の関数の関数記録領域及び戻り先の関数の関数記録領域の間に、障壁を設定された関数記録領域が有る場合に、戻り先の関数記録領域から走査することにより、走査すべき関数記録領域を実質的に少なくすることができるので実時間性を損なうことがなく、しかも実行中の関数の関数記録領域及び戻り先の関数の関数記録領域との間の走査されていない関数記録領域については、参照されることがないので走査を行わなくとも処理異常は発生しない。

[0045] 第６発明に係るコンピュータでの読み取りが可能な記録媒体は、メモリ中のスタック領域に、関数の実行に必要な関数記録領域を積み上げて関数を実行し、実行完了後、当該関数記録領域を破棄する主プログラムを実行するコンピュータに、主プログラムの処理と交互に、関数の実行に使用されるデータを記録するデータ記録領域に記録された保護されていない不要なデータを消去させるプログラムを記録してある、コンピュータでの読み取りが可能な記録媒体において、コンピュータに、スタック領域に積み上げられた関数記録領域を上位側から下位側へ走査させるプログラムコード手段と、コンピュータに、走査された関数記録領域に記録されているポインタが指示するデータ記録領域のデータの保護を予約する予約マーキ

ングを行わせるプログラムコード手段と、コンピュータに、主プログラムの処理を実行すべく、関数記録領域の走査を中断する場合に、走査が最後に行われた関数記録領域に、関数の実行を制限する障壁を設定させるプログラムコード手段と、コンピュータに、設定された障壁が、関数の実行が完了して破棄されるべき関数記録領域に設定されているとき、関数記録領域の破棄より走査を優先させるプログラムコード手段とを含むコンピュータプログラムを記録してあることを特徴とする。

[0046] 第６発明に係るコンピュータでの読み取りが可能な記録媒体では、記録されているプログラムを、主プログラムを処理するコンピュータにて実行することで、コンピュータがデータ消去装置として動作するので、走査済みの関数記録領域に障壁を設定して、障壁が設定された関数記録領域が破棄されるときに、主プログラムの処理より走査及び予約マーキングを優先して行うことにより、関数記録領域が破棄されてカレントフレームが未走査の関数記録領域に移行することを防止するので、走査処理を中断している間の主プログラムの実行に基づくポインタの変化により発生する取りこぼしを防止して、必要なデータの消去を回避し、処理異常の発生を防止して、主プログラム処理時の実時間性を損なわずにデータ記録領域に記録された不要なデータの消去が可能である。

Brief Description of Drawings

[0047] 図１は本発明のデータ消去装置の構成を示すブロック図である。

[0048] 図２は本発明のデータ消去方法を概念的に示す説明図である。

[0049] 図３は本発明のデータ消去方法における障壁を概念的に示す説明図である。

[0050] 図４は本発明のデータ消去方法を概念的に示す説明図である。

[0051] 図５は本発明のデータ消去方法の処理手順を示すフローチャートである。

[0052] 図６は本発明のデータ消去方法の処理手順を示すフローチャートである。

[0053] 図７は本発明のデータ消去方法の処理手順を示すフローチャートである。

[0054] 図８は本発明のデータ消去方法を概念的に示す説明図である。

[0055] 図９は本発明のデータ消去方法の処理手順を示すフローチャートである。

[0056] 図１０は主プログラムの実行に必要な関数の関数記録領域を概念的に示す説明図である。

[0057] 図１１は従来のデータ消去方法の走査及びマーキング処理を概念的に示す説明図である。

[0058] 図１２は従来のデータ消去方法の走査及びマーキング処理を概念的に示す説明図である。

[0059] 図１３は従来のデータ消去方法の走査及びマーキング処理を概念的に示す説明図である。

[0060] 図１４は従来のデータ消去方法の走査処理を概念的に示す説明図である。

[0061] 図１５は従来のデータ消去方法の走査処理を概念的に示す説明図である。

Detailed Description

[0062] 以下、本発明をその実施の形態を示す図面に基づいて詳述する。図１は本発明のデータ消去装置の構成を示すブロック図である。図中１０は本発明の汎用コンピュータを用いたデータ消去装置であり、データ消去装置１０は本発明のデータ消去装置用のプログラム及びデータ等の情報を記録したＣＤ－ＲＯＭ等の記録媒体２０からプログラム及びデータ等の情報を読み取るＣＤ－ＲＯＭドライブ等の補助記憶手段１２、補助記憶手段１２により読み取られたプログラム及びデータ等の情報を記録するハードディスク等の磁気記録手段１３、並びに各種情報を一時的に記録するメモリ手段１４を備えている。

[0063] また磁気記録手段１３には、本発明のデータ消去装置用のプログラム及びデータ等の情報以外にも、Ｊａｖａ、Ｃ＋＋、Ｌｉｓｐ及びＰｒｏｌｏｇ等の動的にデータを生成するプログラミング言語を用いて作成された主プログラムが記録されている。なお本発明のプログラムは、主プログラムを実行するためのプログラムの一部として記録されていてもよい。

[0064] そして磁気記録手段１３からプログラム及びデータ等の情報を読み取り、メモリ手段１４に記録して、ＣＰＵ１１により実行することで、汎用コンピュータは、本発明のデータ消去装置１０として動作する。

[0065] なおメモリ手段１４の記録領域の一部は、主プログラムの処理の実行に必要な関数についての関数記録領域が確保されるスタック領域及び関数の処理の実行に必要なデータを記録するデータ記録領域として用いられる。

[0066] さらにデータ消去装置１０は、マウス及びキーボード等の入力手段１５、並びにモニタ及びプリンタ等の出力手段１６を備えている。また主

プログラムが、例えば各種機械を実時間制御するプログラムである場合、データ消去装置１０は、更に通信手段１７を備え、通信手段１７を介して制御すべき機械に接続される。

[0067] 次に本発明のデータ消去方法の処理内容を説明する。なお本発明のデータ消去方法は、従来のスナップショットＧＣを応用した方法である。図２は、本発明のデータ消去方法を概念的に示す説明図である。なお図２において、左下がりの斜線部は、走査済みの関数記録領域、そして右下がりの斜線部は、未走査の関数記録領域を示し、太枠で囲まれた部分がカレントフレームを示している。

[0068] 図２（ａ）に示すように、本発明のデータ消去方法では、走査を中断する場合、走査が最後に行われた関数記録領域に障壁を設定する。設定された障壁は、主プログラムの処理を制限する目的で設定されるものであり、カレントフレームにある関数の実行が完了して当該関数記録領域を破棄する場合に、破棄すべき関数記録領域に障壁が設定されているとき、図２（ｂ）に示すように、関数記録領域を破棄する処理を中断して、関数記録領域の走査を一時的に行うことで、関数記録領域の破棄により、未走査のスタック領域にカレントフレームが移動することを防止し、そして走査完了後、図２（ｃ）に示すように当該関数記録領域の破棄を行い、カレントフレームを移動させる。

[0069] このように設定された障壁は主プログラムの処理の一つである関数記録領域の破棄処理より、走査処理を優先させるものであり、これによって走査と共に行われる予約マーキングの取りこぼしを防止することが可能である。

[0070] 図３は、本発明のデータ消去方法における障壁を概念的に示す説明図である。図３（ａ）は、障壁が設定される前のカレントフレーム内の関数に含まれるデータ群を示しており、図３（ｂ）は、障壁が設定されたカレントフレーム内の関数に含まれるデータ群を示している。

[0071] 本発明のデータ消去方法では、図３（ａ）及び（ｂ）に示すように、関数に含まれるデータ群中の戻り番地(return addr) を、走査を優先して行わせる関数(Scan frames and return to:)の番地に置換することにより、主プログラムの実行を制限する障壁としている。なお障壁の解除処理は、置換された番地を元に戻すことにより行われる。

[0072] このため主プログラムの変更等の作業が不要であり、しかも例えば障壁の存在を検出するというような処理も不要であるため、オーバーヘッドの発生を抑制することが可能である。

[0073] 図４は、本発明のデータ消去方法を概念的に示す説明図である。図４では、主プログラムにより実行する関数が以下の Common Lisp コードを含む局所関数である場合を示している。

[0074]
```
(defun F(x)
        (labels((G(y z)...(G z y)...))
                (G x x)))
```

[0075] 即ち関数Ｆの中で、局所関数Ｇが定義されており、関数Ｇからは関数Ｇ自身が再帰的に呼び出されており、また再帰呼び出しの式に現れる変数ｘは、関数Ｆにて定義されているパラメータであり、関数Ｆの関数記録領域中に存在する。このため関数Ｇの実行中は、関数Ｇの関数記録領域だけでなく、該関数記録領域より下位の関数Ｆの関数記録領域をも参照する

ことになり、関数Ｇの関数記録領域に、関数Ｆの関数記録領域の参照を可能とする静的リンクを格納することにより実現している。

[0076] 本発明では、局所関数をサポートする言語システムに対応させるため、図４に示すように関数Ｇの関数記録領域を走査する場合、関数Ｇが関数Ｆを実行する下位の関数記録領域を参照しているか否かを判定し、下位の関数記録領域を参照していると判定したときに、関数Ｇの関数記録領域を走査する際に、関数Ｆを実行する下位の関数記録領域をも走査する。このとき関数Ｇの関数記録領域及び関数Ｆの関数記録領域の間の関数記録領域の走査は、関数Ｆの関数記録領域の走査を行った後に行われることになる。なお関数Ｆから更に下位の関数記録領域を参照している場合には、その関数記録領域についても走査を行う。

[0077] 次に本発明のデータ消去方法の処理手順を図５、図６及び図７に示すフローチャートを用いて説明する。データ消去装置１０では、主プログラムの関数の処理と交互に、不要なデータを消去する処理を実行する。データ消去の処理手順としては、主プログラムが中断している間に、スタック領域に積み上げられている関数記録領域を、上位側から下位側へ走査する処理を行い（Ｓ１０１）、走査された関数記録領域に記録されているポインタが、直接的又は間接的に指示するデータ記録領域のデータの保護を予約する予約マーキングを行う（Ｓ１０２）。

[0078] なお主プログラムが図４を用いて説明したように局所関数をサポートする言語システムに対応している場合には、ステップＳ１０１の関数記録領域走査処理において、図６に示すように静的リンクによる参照先の有無を判定する処理が行われる。即ち走査する関数記録領域（以降第１関数記録領域という）にて実行される関数が、第１関数記録領域より下位の関

数記録領域（以降第２関数記録領域という）を参照しているか否かを判定し（Ｓ２０１）、第２関数記録領域を参照していると判定した場合（Ｓ２０１：ＹＥＳ）、ステップＳ１０１による関数記録領域走査処理として、第１関数記録領域だけでなく第２関数記録領域をも走査する（Ｓ２０２）。そしてステップＳ１０２に進み予約マーキング処理を行う。なおここでいう第１関数記録領域にて実行される関数とは、図４における関数Ｇに相当し、第２関数記録領域にて実行される関数とは、図４における関数Ｆに相当する。

[0079] またステップＳ２０１において、第２関数記録領域を参照していないと判定した場合（Ｓ２０１：ＮＯ）、ステップＳ１０１による関数記録領域走査処理として、第１関数記録領域のみを走査し（Ｓ２０３）、ステップＳ１０２に進み予約マーキング処理を行う。

[0080] そして主プログラムの処理を再開すべく、ステップＳ１０１及びＳ１０２の処理を中断する場合に、最後に走査を行った関数記録領域に、関数の実行を制限する障壁を設定する（Ｓ１０３）。

[0081] 障壁設定後、主プログラムの関数を実行し（Ｓ１０４）、カレントフレームが、障壁を設定してある関数記録領域に移行した場合で、当該関数の実行が完了して当該関数記録領域を破棄するときに、破棄処理より走査を優先し（Ｓ１０５）、また走査に伴い予約マーキングを行う（Ｓ１０６）。なおこのときも局所関数をサポートする言語システムに対応している場合については、図６に示す処理が行われることは言うまでもない。そして関数の実行、即ち関数記録領域の破棄を再開する（Ｓ１０７）。

[0082] またステップＳ１０５～Ｓ１０７の処理は、具体的には、図３（ｂ）に示すように、走査（及び予約マーキング）を行わせた後、主プロ

グラムの関数を実行させる関数を呼び出すために、障壁として設定した番地へ制御を移行することを示す。

[0083] そして走査完了後、主プログラムが中断している間に、図７に示すように不要なデータを消去する処理として、予約マーキングされているデータ記録領域に、マーキングを行い（Ｓ３０１）、マーキングされていないデータ記録領域のデータを消去する（Ｓ３０２）。

[0084] 次に本発明のデータ消去方法において、主プログラムの実行の中断等の割込命令による例外処理又は非局所的脱出により、実行する関数が、実行中の関数から、該関数を呼び出した関数ではない下位の戻り先となる関数となる状況が発生した場合の処理、例えば非局所的脱出をサポートする言語システムに対応させる場合に実装すべき処理を説明する。

[0085] 図８は、本発明のデータ消去方法を概念的に示す説明図である。図８では、図８（ａ）に示すように関数Ｈを実行している状態から、非局所的脱出が行われることにより、関数Ｈの実行を終了させた後、図８（ｂ）に示すように関数Ｈを呼び出した関数より更に下位の関数Ｆを戻り先の関数として実行するthrowと呼ばれる機能が実行される状況を示している。

[0086] 本発明では、非局所的脱出をサポートする言語システムに対応させるため、図８に示すように戻り先の関数Ｆが、実行中の関数Ｈの関数記録領域を積み上げさせた関数と異なる更に下位の関数Ｆである場合で、実行中の関数Ｈの関数記録領域及び戻り先の関数Ｆの関数記録領域の間に障壁を設定された関数記録領域が有るときに、設定されている障壁を解除して、戻り先の関数記録領域から走査の再開、予約マーキング及び障壁の設定を行う。

[0087] このため関数Hを実行していた関数記録領域及び関数Fを実行する関数記録領域の間に、走査されない関数記録領域が生じることになり、該関数記録領域に記録されているポインタから直接又は間接的に指示されているデータ記録領域のデータは、予約マーキングされないため消去されることになるが、それらのデータは参照されることが無いので異常処理は発生しない。

[0088] 次に本発明のデータ消去方法の処理手順を図９に示すフローチャートを用いて説明する。データ消去装置１０では、非局所的脱出が行われる場合、先ず関数の関数記録領域から下位側へ向かって、戻り先となる関数を探す処理が行われる。このとき戻り先関数を探す処理とともに障壁を設定された関数記録領域の有無を判定する。そして実行中の関数の関数記録領域及び戻り先の関数の関数記録領域の間に障壁を設定された関数記録領域が有ると判定した場合（Ｓ４０１：ＹＥＳ）、設定されている障壁を解除し（Ｓ４０２）、戻り先の関数の関数記録領域から走査、予約マーキング及び障壁の設定を開始する（Ｓ４０３）。

[0089] ステップＳ４０１において、実行中の関数の関数記録領域及び戻り先の関数の関数記録領域の間に障壁を設定された関数記録領域が無いと判定した場合（Ｓ４０１：ＮＯ）、戻り先の関数の関数記録領域が既に走査されていることになるので、ステップＳ４０２及びＳ４０３の処理は行われない。

[0090] 前記実施の形態では、各種機械を実時間制御すべく、本発明のデータ消去装置となる汎用コンピュータを接続する形態を示したが、本発明はこれに限らず、実時間制御が必要な各種機械を備える制御用マイクロコン

ピュータに、本発明のデータ消去方法を適用するプログラミングをしてもよい。

[0091] 以上詳述した如く本発明に係るデータ消去方法、データ消去装置及び記録媒体では、Ｊａｖａ、Ｃ＋＋、Ｌｉｓｐ及びＰｒｏｌｏｇ等の動的にデータを生成するプログラミング言語を用いて作成された主プログラムの処理と交互に実行される不要なデータの消去処理として、走査済みの関数記録領域に障壁を設定して、障壁が設定された関数記録領域が破棄されるときに、主プログラムの実行より走査及び予約マーキングを優先して行うことにより、関数記録領域が破棄されてカレントフレームが未走査の関数記録領域に移行することを防止するので、走査処理を中断している間の主プログラムの実行に基づくポインタの変化により発生する予約マーキングの取りこぼしを防止して、必要なデータの消去を回避し、処理異常の発生を防止して、主プログラム実行時の実時間性を損なわずにデータ記録領域に記録された不要なデータの消去が可能である等、優れた効果を奏する。

[0092] さらに本発明では、走査を行わせる関数の番地を障壁として設定することにより、関数記録領域の破棄より走査を優先させる処理のための主プログラムの修正が不要であり、しかもその処理によるオーバーヘッドを抑制するので、主プログラムの実行時の実時間性を損なわずにデータ記録領域に記録された不要なデータの消去が可能である等、優れた効果を奏する。

[0093] また本発明では、局所関数をサポートする言語システムの場合において、走査すべき関数記録領域にて実行される関数が、下位の関数記録領域を参照しているときに、参照元の関数記録領域だけでなく参照先の関数記録領域をも走査することにより、例えば障壁が参照元の関数記録領域及

び参照先の関数記録領域の間に設定されていたとしても処理異常の発生を防止し、しかも走査すべき関数記録領域を実質的に少なくすることができるので実時間性を損なうことがない等、優れた効果を奏する。

[0094] そして本発明では、実行する関数が、実行中の関数から、該関数を呼び出した関数ではない更に下位の戻り先となる関数に移行する throw が実行される場合において、実行中の関数の関数記録領域と戻り先となる関数の関数記録領域との間に障壁が設定されているときに、戻り先となる関数の関数記録領域から走査を開始することにより、走査すべき関数記録領域を実質的に少なくすることができるので実時間性を損なうことがなく、しかも実行中の関数の関数記録領域及び戻り先となる関数の関数記録領域との間の走査されていない関数記録領域については、参照されることがないので走査を行わなくとも処理異常は発生しない等、優れた効果を奏する。